Atomic Design 入門

はじめに

Atomic Designをどうやってやっていけばいいの？
って人向けの資料

発表者自体も手探りでやっている部分があるのでご了承を。。

Atomic Designとは

デザイン要素を 原子（Atoms）と見立てて、組み合わせることで
より大きなパターンやコンポーネントを作り出す方法論。

Atomic Designは、デザインを階層的に組織し、管理可能で再利用可能な部品に分割することを目的としている。

Atomic Designは通常、5つの構成で説明されます。

提唱者が説明しているサイト: https://atomicdesign.bradfrost.com

Atomic Designの構成要素

以下の5つの要素を組み合わせて作成する

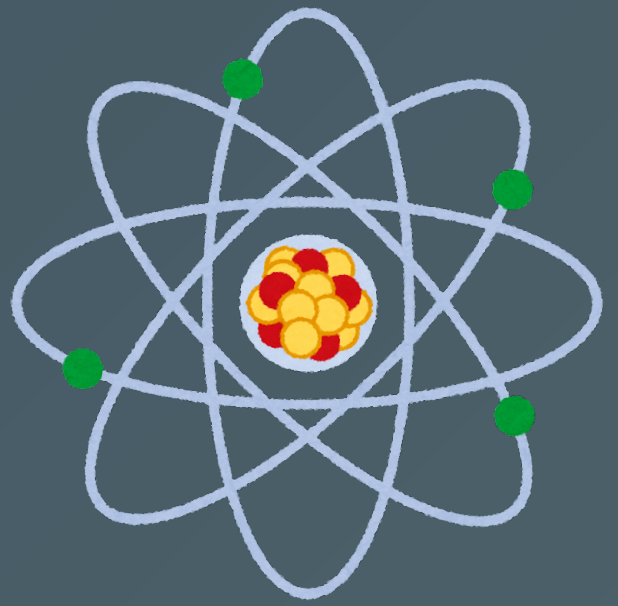
- Atoms : 原子に該当
- Molecules : 分子に該当
- Organisms : 有機体に該当
- Templates : 画面のレイアウト自体に該当
- Pages : 画面本体に該当

Atoms

"Atoms"（原子）はAtomic Designの最も基本的な要素であり、デザインの中で最も小さな単位でそれ以上分解できないもの。

Atoms は通常、一貫性を持たせるために使用され、再利用性が高い特性を持つべきものとなっている。

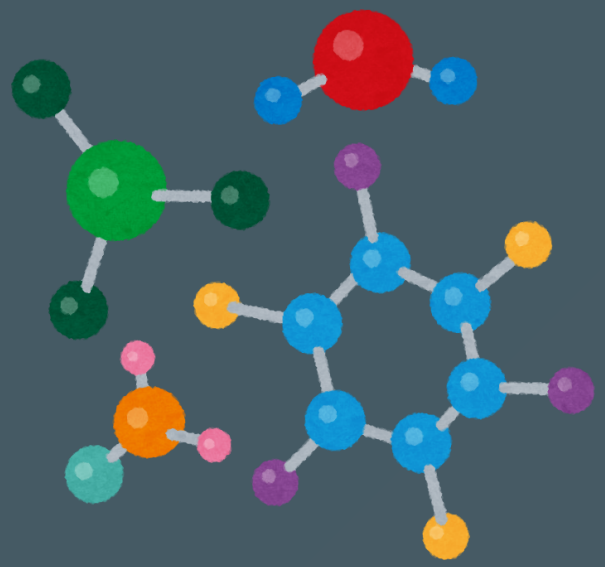
例: `<button>`, `<input>`, `<div>` などのHTMLタグ



Atoms Sample

コンポーネントの例。
基本的にはHTMLタグと一対一の対応関係になる

```
const Button = ({ onClick, children }) => {
  return (
    <button onClick={onClick}>
      {children}
    </button>
  );
};
```



Molecules

"Molecules"（分子）は、個々の要素を組み合わせてできた機能的で独立したもの。異なる原子（Atoms）を結合させて特定の機能やタスクを実現するためのコンポーネント。

Moleculesは、単独で存在するだけでなく、他のMoleculesやAtomsとも組み合わせて使用されることがあります。

例: 検索バー、アイコン付きボタンなど

Molecules Sample

コンポーネントの例。
Atomsのサンプル(Button)を使ってMoleculesを作成しています。

```
import Button from '../atoms/Button'

const IconButton = ({ icon, label, onClick }) => {
  return (
    <Button onClick={onClick}>
      <span className="icon">{icon}</span>
      <span className="label">{label}</span>
    </Button>
  );
};
```

Organisms

"Organisms"（有機体）は、Atoms（原子）、分子（Molecules）を組み合わせたもの。

Organismsは、ユーザーが利用する上で単一の機能や目的を果たす機能的な単位を形成します。

例: ヘッダー、フッター、カード、フォーム等

Organisms Sample

コンポーネントの例。
Atoms, Moleculesを組み合わせてヘッダーを作成しています。

```
import Label from '../atoms/Label'
import LinkList from '../molecules/LinkList'

const Header = () => {
  return (
    <header>
      <Label name='My Website' style='h1'/>
      <LinkList list={{name: 'Home', path: '/home'}, {name: 'About', path: '/about'}}/>
    </header>
  );
};
```

Templates

テンプレート（Templates）はAtomic Designの階層の中で、有機体（Organisms）やそれ以下の階層のコンポーネントを配置し、ページの骨組みや構造を表現するもの。

Templates Sample

Atoms, Molecules, Organismsを組み合わせて作成。

```
import Card from '../molecules/Card'
import Table from '../molecules/Table'

const Template = ({contents, data}) => (
  <div>
    {
      contents.map((row) => {
        <Card title={row.title} detail={row.detail}/>
      })
    }
    <TableData data={data}>
  </div>
);
```

Pages

Pages（ページ）は、Atomic Designの階層構造における最上位の要素であり、実際のコンテンツが配置される具体的なウェブページやアプリケーション画面になる。

実際に使用するデータなどはここで取得し、Templeteに値を渡すことで表示する。

Pages Sample

取得したデータをTempletesの `props` に渡して表示するようにしている

```
import Card from '../templetes/Templete'
const [data, setData] = useState([]);

useEffect(() => {
  fetch('https://example.com/data')
    .then(response => response.json())
    .then(data => setData(data))
    .catch(error => console.error('データの取得に失敗しました', error));
}, []);

const Page = ({}) => (
  <Templete data={data}>
);
```

分類

あまり厳密ではないが、対応関係としては

Pages > Templetes > Organisms > Molecules > Atoms

基本上位から下位を呼んではいけないことは前提になります

分類

Atoms から Atoms を呼んではいけない

Molecules から Molecules は呼んでいいし、Atoms を呼んでもいい

Organisms は Atoms, Molecules 呼んでOK
Organisms から Organisms を呼ぶのはあまり推奨しない
（あまりに大きいものになる場合は許容してもいいかも）

Templates は Atoms, Molecules, Organisms を呼んでよい

Pages は Templates を呼ぶだけになるべくするべきだが、
Atoms, Molecules, Organismsを呼んでもよい（はず）

Logic

処理（業務ロジック）を書く場所は分離してもいいし、
書くなら Pages に書く形になるはず

例えば、Redux の Action だったり、Component 内の関数だったり

業務と密接に繋がっている Component だった場合は Organisms とかに処理を書いてしまってもいいと思います

あとは、テストのし易さとコーディングの煩雑さを天秤にかけて判断

デザイン管理

Atomic Designを使用すると
同じような物を作ってしまうことが多々ありそう

その場合は、Storybook なるものを使うとよさそうです

簡単に言うと「UIカタログ」みたいなもの
Component をプレビューで確認できる便利なやつみたいです

https://storybook.js.org

終わり

ご清聴ありがとうございました